Quizz 4 - Rails

Full name

Q1 - How do you create a Rails app?

```
➜
```

Q2 - How do you start coding a Rails project? Give the right sequence.

1. Coding the views?
2. Coding the controllers?
3. Coding the models?

Q3 - How do you generate a `Song` model with a `title` and a `year`?

```
➜
```

What are the 2 created files?

What is the **rails** command you should type then?

```
➜
```

Q4 - How do you add a `category` (ex: `"rock"`, `"electro"`, etc..) to your songs table using the correct Rails generator?

```
➜
```

What is the created file?

What is the **rails** command you should type again?

```
➜
```

Q5 - Add a validation on the presence of a song title & crash-test your model in the console

```ruby
# models/song.rb
class Song < ApplicationRecord
  # Add the validation

end
```

Now crash-test your model:

```
➜ rails c
pry>
pry>
pry>
pry>
pry>
pry>
pry>
```

Q6 - What is the Rails flow you need to follow again and again? Give the correct order

1. The Router is routing the HTTP request to "controller#action"
2. The action is getting data from models
3. Everything starts with an HTTP Request
4. The action is rendering the view

Q7 - What are the **4 different parts** inside an HTTP request?

1.
2.
3.
4.

Q8 - Are the HTTP requests in the following 2 routes the same? Why?

```ruby
# config/routes.rb
get "/songs" => "songs#index"
post "/songs" => "songs#create"
```

Q9 - What's the difference between a **GET** and a **POST** request?

Q10 - Complete the controller code using the correct `params` key?

HTTP request:

```
GET /search?query=thriller
```

Routing:

```ruby
# config/routes.rb
get "/search" => "songs#search"
```

Controller:

```ruby
class SongsController < ApplicationController
  def search
    # TODO
    @songs =
  end
end
```

Q11 - Complete the controller code using the correct `params` key?

HTTP request:

```
GET /songs/named/thriller
```

Routing:

```ruby
# config/routes.rb
get "/songs/named/:name" => "songs#search"
```

Controller:

```ruby
class SongsController < ApplicationController
  def search
    # TODO
    @songs =
  end
end
```

Q12 - What are the 7 CRUD routes generated by the `resources` method in Rails?

```ruby
# config/routes.rb
# TODO: Give us the details of the 7 routes generated with:
resources :songs
# HINT: HTTP-verb "url" => "controller#action"







```

Q13 - How do you print your routes and their URL prefix helpers?

```
➜
```

Q14 - How do you generate a controller for your songs?

```
➜
```

Q15 - Implement the **R**ead actions in your songs controller?

```ruby
class SongsController < ApplicationController






end
```

Q16 - What are the 2 requests needed to create a new song? Implement the `songs#new` and `songs#create` actions.

```ruby
class SongsController < ApplicationController
  def new

  end
  def create







  end

  private
  def song_params

  end
end
```

Q17 - Why do we have to filter parameters using "strong params" in the controller?

Q18 - **Hard question**: What is the HTML generated?

```ruby
@song = Song.new
```

Now what is the HTML code generated by:

```erb
<%= form_for @song do |f| %>
  <%= f.text_field :title %>
  <%= f.submit %>
<% end %>
```

Fill the blanks:

```html
<form action="                    " method="post">
```

```
  <input type="text" name="           " value="           ">
  <input type="submit" value="Create song">
</form>
```

Q19 - **Hard question**: What is the HTML generated?

Imagine that:

```
@song # => <#Song: id: 18, title: "Hey jude", year: 1968, category: "rock">
```

Now what is the HTML code generated by:

```
<%= form_for @song do |f| %>
  <%= f.text_field :title %>
  <%= f.submit %>
<% end %>
```

Fill the blanks:

```
<form action="                    " method="patch">
  <input type="text" name="              " value="              ">
  <input type="submit" value="Create song">
</form>
```

Adding a 2nd model

Q20 - Now you want to add reviews to your app. Here are some constraints

- We don't want our visitors to destroy or update reviews, just to create ones.
- We don't want a separate index page to list all reviews or a show page to display each review. Instead, we want to display reviews on the show page of each song, for better UX.

Step #1: Model

Generate your `Review` model in the terminal. It should have only a `content:string` and a `song:references` (= the foreign key).

```
➜
```

Run the migration

```
➜
```

Add validation/associations

- Add a validation for the presence of a content
- Add associations between `Review` and `Song`

```
class Song < ApplicationRecord

end
```

```
class Review < ApplicationRecord


end
```

Step #2: Routing/Controller

Generate the reviews controller

```
➜
```

Add the **necessary** routes (don’t forget **we don’t want the 7 CRUD actions for reviews**)

```ruby
# config/routes.rb
resources :songs do
  # TODO

end
```

Now code your controller:

```ruby
class ReviewsController < ApplicationController
  before_action :set_song

  def new

  end

  def create




  end

  private
  def set_song
    @song = Song.find(params[:song_id])
  end
  def review_params
    params.require(:review).permit(:content)
  end
end
```

Step #3: Views

Add a song’s reviews on its show page:

```erb
<h1><%= @song.title %></h1>
<p><%= @song.year %></p>
<p><%= @song.category %></p>
<h2>Here are the reviews for this song:</h2>




```